Capítulo 5 – Princípios de Projeto (Resolução Completa)

1. Quais os benefícios do Ocultamento de Informação?

Três principais benefícios:

1. **Desenvolvimento em paralelo:** diferentes desenvolvedores podem trabalhar em partes distintas do sistema sem interferência, desde que conheçam apenas as interfaces públicas.

2. **Flexibilidade a mudanças:** detalhes internos podem mudar sem afetar outras partes do código.

3. **Facilidade de entendimento:** cada módulo precisa compreender apenas as classes com as quais interage.

2. O que ocorre se várias classes forem movidas para o mesmo arquivo?

Melhora: Coesão – classes relacionadas ficam próximas.

Prejudica: Ocultamento de informação – aumenta o acoplamento e reduz o isolamento entre módulos.

3. O que é 'classitis' e por que é ruim?

É o excesso de classes muito pequenas e fragmentadas.

Isso gera **acoplamento excessivo** e aumenta a **complexidade global**, tornando o sistema mais difícil de entender e manter.

4. Quais são os tipos de acoplamento?

- **Aceitável:** depende apenas da interface pública.

- **Ruim:** depende de detalhes internos de outra classe.

- **Estrutural:** dependência explícita entre classes.

- **Evolutivo:** dependência indireta entre classes que mudam juntas.

5. Dê exemplos de acoplamento estrutural aceitável e ruim.

Aceitável: uso de classes da biblioteca padrão (ex: `Hashtable`).

Ruim: acesso direto a atributos internos de outra classe.

6. Pode haver acoplamento sem referência explícita?

Sim. Por exemplo, quando duas classes compartilham um arquivo ou variável global.

Esse é o **acoplamento evolutivo**, considerado perigoso.

7. Qual problema há em colocar todo o código no método main?

Baixa **coesão**. O método concentra múltiplas responsabilidades (entrada, lógica, saída), violando o princípio da responsabilidade única.

8. Que princípio é violado no código onclick()?

Viola o **Princípio da Responsabilidade Única** e o **Princípio de Demeter**.

Mistura responsabilidades de interface, negócio e persistência.

9. Se fosse preciso eliminar um conceito OO, qual seria?

A **herança**, pois pode ser substituída por composição. Encapsulamento e polimorfismo são fundamentais para modularidade.

10. O que há de errado no método sendMail?

Viola o **Princípio de Demeter** ('não fale com o amigo do amigo').

A responsabilidade de enviar e-mail deve estar em `ContaBancaria`.

11. E no método imprimeDataContratacao()?

Também viola o **Princípio de Demeter**.

Crie um método `getDataContratacaoFormatada()` dentro de `Funcionario`.

12. O que está errado com as pré e pós-condições da subclasse?

Viola o **Princípio de Substituição de Liskov (LSP)**.

A subclasse não pode tornar pré-condições mais restritas nem pós-condições mais fracas.

13. Qual o CBO e o LCOM da classe A?

CBO(A) = 5 (B, C, D, E, F)

LCOM = 1 (um par de métodos sem atributos em comum).

14. Qual classe é mais coesa?

Classe A: LCOM = 0

Classe B: LCOM = 3

→ A é mais coesa, pois todos os métodos compartilham o mesmo atributo.

15. Por que LCOM mede ausência e não presença de coesão?

Porque ele **conta os pares de métodos que não compartilham atributos**, ou seja, mede a falta de coesão.

16. Todos os métodos devem contar no LCOM?

Não. **Construtores e getters/setters** devem ser ignorados, pois distorcem o resultado.

17. Complexidade ciclomática depende da linguagem?

Não. Mede apenas a **estrutura de decisão lógica** (if, while, for, etc.), independente da sintaxe.

18. Dê um exemplo com complexidade mínima.

```java
void exibeMensagem() {
System.out.println('Olá!');
}
```

Complexidade ciclomática = 1.

19. Compare versões monolítica e OO.

Versão monolítica:

- Mistura responsabilidades em um só bloco.
- Dificulta testes e manutenção.

Versão OO:

- Divide responsabilidades em classes.
- Facilita testes e manutenção.

- Reduz acoplamento e aumenta coesão.